

Snap2Pass: Consumer-Friendly Challenge-Response Authentication with a Phone

Ben Dodson, Debangsu Sengupta, Dan Boneh, and Monica S. Lam*

Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

This paper proposes a challenge-response authentication system called Snap2Pass that is easy to use and provides strong security guarantees. The system uses QR codes which are small two-dimensional pictures that encode digital data. When logging in to a site, the web server sends the PC browser a QR code that encodes a cryptographic challenge; the user takes a picture of the QR code with his cell phone camera which results in a cryptographic response sent to the server; the web server then logs the PC browser in. Our user study shows that authentication using Snap2Pass is easy to learn and considerably faster than existing one-time password and challenge-response systems. By implementing our solution as an OpenID provider, we have made this scheme available to over 30,000 websites that use OpenID today.

Snap2Pass is based on Snap2, a general mechanism that uses QR codes to establish a secure connection between a server, a PC browser and a cell phone quickly. Based on this mechanism, we demonstrate how to securely pair the cell phone directly with a web browser with a simple snap of the QR code. We can then operate the phone from the PC browser window to make phone calls or send SMS messages. We can also cut-and-paste across the phone and the PC browser, making it easy to type long text passages onto the phone.

1. Introduction

Passwords are the predominant form of authentication system used by today's websites. It is not because the password system is secure; quite the contrary, they are known to have many problems. Passwords are vulnerable to dictionary attacks and can be easily phished using a spoofed web site. Moreover, since users tend to use the same password at many sites, a single server compromise can result in account takeover at many other sites. Despite these limitations passwords are widely used.

Over the years, many enhancements have been proposed, including smart cards, one-time password tokens (such as RSA SecurID) and challenge-response authentication. To date, none of these have been widely adopted on the Web.

Challenge-response is a good case study. While it prevents some attacks that defeat basic passwords, it is rarely used on the Web due to the cumbersome user experience. For example, a system called CRYPTOCard uses a smart-card with a screen and a keyboard where users key in the challenge and then copy the response to the desktop. Authentication using CRYPTOCard takes far longer than authentication using a simple password. As a result, CRYPTOCard is primarily used in corporate settings where the additional hardware cost and the extra inconvenience is acceptable.

This paper introduces a new technique called Snap2Pass that provides a convenient challenge-response system for logging into web sites from a PC. Snap2Pass requires no special hardware beyond a cell phone with a camera. There is no more memorization of passwords and the login process is faster and less error prone than with existing systems such as one-time passwords. Our initial user study confirms the Snap2Pass ease of use and speed.

1.1 Fast Secure Login on the Web Browser

The phone is always with us and switched on. It is a personal device—we do not use others' phones, and nobody uses our phone, except in rare circumstances. In fact, the phone is an ideal device for keeping personal, private, information. In other words, it serves to identify the owner, and can be used as a second-factor authentication. Browsers on the PC, on the other hand, are not personal. We often drift between browsers, on different PCs, at home, at work and on the road. Since the PC has a larger screen and a big keyboard, we can make the best of both worlds by pairing the phone as a personal device with a generic PC browser.

At a high level, the user experience using Snap2Pass is as follows. The user navigates his PC browser to the login page of a web site. The login page displays a QR code containing a cryptographic challenge, among other things. The user

* Supported by the NSF POMI project.

takes a picture of the QR code using his cell phone camera. No other user interaction is needed to log in. Under the hood, a pre-shared secret key stored on the phone is used to compute a response to the cryptographic challenge which is then transmitted to the site via the cellular network. The site checks the response and if it verifies, loads the logged-in page on the PC browser through a browser refresh triggered via XMPP. The use of both the phone and PC provides an added security benefit, as checking the co-location of these devices can mitigate man-in-the-middle attacks.

With a snap of the QR code, we have tied our phone to the PC browser, and now we can use these two platforms synergistically to enhance our browsing experience. We can browse on the bigger display of any generic PC available and use the phone to manage our personal identity. There is no setup necessary; we do not have to download any extensions to the browser. Capturing a QR code is easy and fast. Unlike for example pairing with bluetooth, we do not need to read any manuals to learn how to snap a QR code. Consumers already know how to take a picture; they just need to see it done once. The difficulty is identical to swiping a bar code.

1.2 Portaling between the Phone and Web Browser

Our technique to tie the phone and the PC platform together using a QR code has many uses beyond authentication. We refer to this general technique as Snap2. On top of Snap2, we have also developed a system called Snap2Web that leverages the complementary resources of these two platforms to create a better user experience. This system requires no browser configuration.

With Snap2Web, we can walk up to any PC browser, visit a webpage hosting the Snap2Web messaging service, snap the picture of the QR code presented, and the phone is securely paired to the browser. Once the two platforms are tied, we can perform the following:

- *One-click phone calls on the PC.* We can dial a phone number on our phone by clicking a phone number in the PC browser instead of typing the phone number on the phone. The call is placed from the phone in speakerphone mode.
- *SMS on the PC keyboard.* We can send and receive SMS on the PC, without having to type using the phone's keyboard.
- *Sending a web page to the PC.* With a click of a button, we can move a web page from the phone to the PC where we can leverage its larger screen.
- *Cut-and-paste between the PC and the phone.* The cut-and-paste metaphor is now extended across the two platforms. We can use the PC keyboard for entering data onto the phone, and vice versa. The latter is useful for example if we wish to finish a message we were typing in on the phone.

1.3 Contributions

This paper makes the following contributions:

Snap2Pass: A consumer-friendly challenge-response authentication system. This technique is easy to use, requiring users to only take a picture of the QR code with a camera on their cell phones. The website displays a QR code that embeds a challenge. The cell phone sends the response to the challenge directly to the web server.

OpenID implementation. We have implemented a custom OpenID provider that uses Snap2Pass, and a mobile client for the Android environment. This provider can be used immediately to log onto over 30,000 existing websites that use OpenID today [13]. We demonstrate that our techniques can be implemented today with minimal changes to legacy services. No changes are required on browsers.

Snap2Web: Portaling between the phone and the web browser. Snap2Web makes it easy for consumers to leverage the PC resources (large screen, large keyboard) and the phone resources (phone calls and SMS) quickly and seamlessly. We have extended the cut-and-paste metaphor to traverse the two screens—the PC and the phone.

User Study. Our user studies suggest that both Snap2Pass and Snap2Web are easy to use.

The Snap2 technology. Snap2 is a general technique based on the ability to create quickly a secure three-way connection between a server, a PC browser and a phone. The browser connects to the server with a web page visit, which is then connected to the phone via a QR code that embeds a session key. This enables a server to engage in a session with the browser and the phone simultaneously. In Snap2Web, the server acts as a conduit allowing information to pass between the phone and the browser securely. All information is encrypted between the PC and the phone.

1.4 Paper Organization

Section 2 presents the threats addressed by this paper. Section 3 describes the core Snap2Pass algorithm based on both symmetric keys and public/private keys. Next, Section 4 presents a security analysis. Section 5 presents a few deployment models. Section 6 presents the portaling ideas. Section 7 presents our experimental results. We describe our implementation and two user studies, one on Snap2Pass and one on Snap2Web. Section 8 presents related work and Section 9 concludes.

2. Threat model

Snap2Pass is an authentication system designed for ease of use while providing stronger security than traditional passwords or one-time passwords. Our design is intended to protect against the following types of adversaries:

- *Phishing.* Phishing targets users who ignore the information presented in the browser address bar. A phishing attacker sets up a spoof of a banking site and tries to fool the user into authenticating at the spoofed site. Furthermore, we allow for online phishing where the phishing site plays a man-in-the-middle between the real banking site and the user. The phisher can wait for authentication to complete and then hijack the session. One-time-password systems, such as SecurID over SSL, cannot defend against online phishing. With Snap2Pass this attack is considerably harder, as discussed in section 4.2.
- *Network attacker.* We allow the attacker to passively eavesdrop on any network traffic. Moreover, we allow for a wide class of *active* network attackers discussed in Section 4.
- *Phone theft.* Snap2Pass enables quick revocation in case of phone theft.

Snap2Pass does not provide security against malware on the user’s machine. Indeed, a sophisticated transaction generator could, in principle, execute transactions on the user’s behalf once authentication completes. A good example is the stealthy transaction generator described by Jackson et. al [6]. Similarly, Snap2Pass does not protect against malware on the phone itself.

3. The Snap2Pass System

We now present the core algorithms in the Snap2Pass system. Recall that challenge-response authentication comes in two flavors. The first is a system based on symmetric cryptography. It uses little CPU power and generates very short messages, however it requires that the server possess the user’s secret authentication key. As a result, the user must maintain a different secret key for each server where she has an account. The second is a system based on public-key cryptography. It requires considerably more CPU power to generate responses to challenges, but the server only keeps the user’s public-key. Consequently, the same user secret key can be used to authenticate with many servers.

We describe both challenge-response systems as implemented in Snap2Pass. We present the basic work flow, from account creation, login, to revocation.

3.1 Symmetric key Challenge Response Authentication

In a symmetric key based challenge-response system, the client(s) and web server communicate using a pre-shared secret key. Our implementation uses a key length of 128 bits. This key is used in the HMAC-SHA1 algorithm to compute responses to server-issued challenges. The challenges are 128-bit length nonces embedded in a QR code, while the responses are 160 bits long and are sent over the wireless network.

3.1.1 Account Creation

The account creation web page invites a new user to submit a username. Upon receiving an acceptable username, the server generates a shared secret for the account and then sends a QR code to the web page encoding the account information. The user launches the Snap2Pass application on the phone and selects the “Set Account” button to activate the camera, which he uses to consume the QR code. The web site then confirms that the account was created successfully. To avoid adding inaccessible entries in the provider’s database, it should require a user login to complete the creation process.

Figure 1 shows the Snap2Pass application running on the Android phone. We create a QR code representing a created account by encoding the following contents:

```
{ protocol: "V3"
  , provider: "goodbank.com"
  , respondTo:
    "https://login.goodbank.com/response"
  , username: "mr_rich"
  , secret: "2934bab43cd29f23a9ea"
}
```

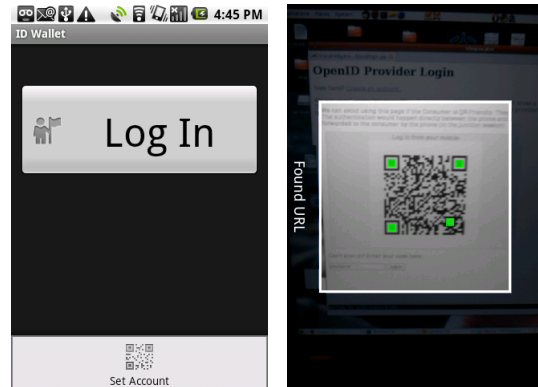


Figure 1. The mobile client running on Android. (a) The home screen, with a single button to log in and with “Set Account” accessible as a menu entry. (b), scanning into a browser session.

Note that this process eliminates the need for a user to create and remember a password, which is not just cumbersome but extremely insecure as discussed in Section 1. As a matter of fact, it is not even necessary for the user to have a friendly user name; however it is important for the sake of addressing and reassuring the user that the website recognizes him.

Instead of using a user-supplied password, our scheme allows the web server to generate a random key as a shared secret between the web site and the user. The shared secret is presented in a QR code and saved on the phone’s password manager once scanned. The user can present it for subsequent logins without needing to know its value.

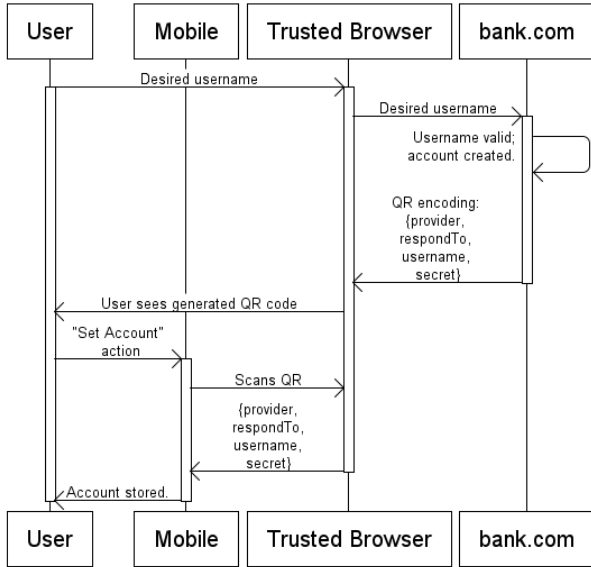


Figure 2. A sequence diagram for creating an account in Snap2Pass.

The QR code also specifies the endpoint where the phone will send responses to challenges as part of the login procedure. A sequence diagram showing the account creation protocol is shown in Figure 2.

In principle, account creation can be done entirely on the phone, without the need for an interaction between the PC and the phone. We chose not to use this approach in Snap2Pass since during account creation the user is often required to supply account details such as a physical address, email address, etc. Typing all this information on the phone can be cumbersome. Instead, with Snap2Pass the user enters all account details on the PC and uses a QR code to move the corresponding credentials to the phone.

3.1.2 Account Login

On the login page, a website displays a QR code and asks the user to snap the picture with his phone's camera to log in. Figure 3 shows a mock up of what a Gmail login screen would look like using Snap2Pass. Note that the page provides an alternative login method in case the user's phone is not available.

The QR code on the login page, unique per session, encodes a random, challenge nonce to be used in the symmetric challenge-response authentication. This is generally presented within the context of an SSL session between the browser and the web server. An example of the contents contained in a challenge QR code shown at the time of login:

```
{
  protocol: "V3"
  , provider: "goodbank.com"
  , challenge: "59b239ab129ec93f1a"
}
```



Figure 3. A mock up of a Gmail log-in site using a QR code.

By binding the challenge nonce to the browser session, the server ensures that only one browser session can make use of its authorization.

To log in, the user launches the mobile Snap2Pass application and selects "Log In". By using the phone's camera, the application consumes the challenge QR code and extracts the challenge within. The application finds a shared secret key and response endpoint that match the provider name and desired user account. It computes a response comprising of the HMAC-SHA1 hash of the entire challenge message using the pre-shared secret as key and sends it to the response endpoint, as well as the original challenge and account identifier. The provider verifies this response and, if successful, the browser session is authenticated with the appropriate account. An example of a response message is shown below:

```
{
  protocol: "V3"
  , challenge: "59b239ab129ec93f1a"
  , response: "14432nafdrwe2443af"
  , username: "mr_rich"
}
```

The challenge and response flows occur within SSL sessions. A sequence diagram showing the login protocol is shown in Figure 4.

3.2 Public-key Based Challenge Response Authentication

Key proliferation is a prevalent problem with a challenge response system that utilizes symmetric keys. The user needs to negotiate and manage a shared secret with each web site he visits. We describe a public-key based challenge response systems to combat this problem.

Instead of using symmetric keys, the Snap2Pass mobile application can generate private/public key pair for the user

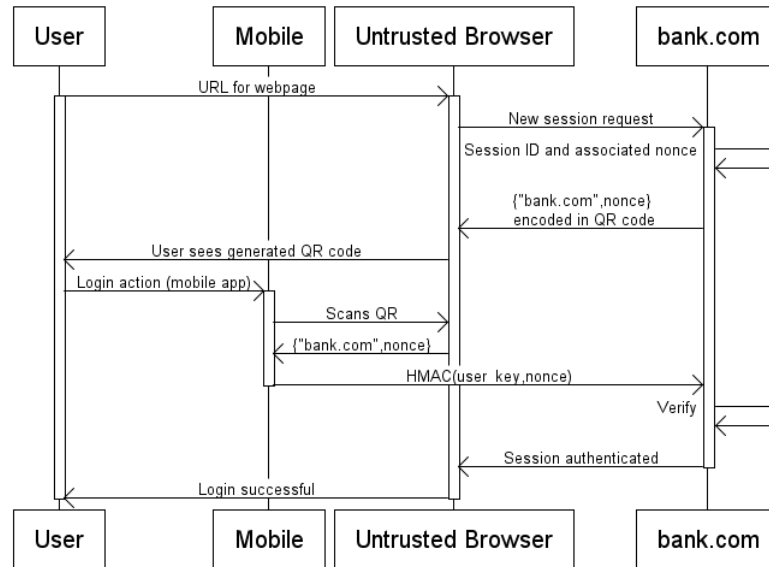


Figure 4. A sequence diagram for logging in to a web application using Snap2Pass.

upon installation (and on-demand). The account creation step is modified such that the user presents his public key to the web site instead of having the site generate it. The challenge process proceeds as before. The Snap2Pass application generates a response by signing the challenge with the private key. The web server verifies the response by matching it against the user’s public key.

The user’s public key can be used across all the sites that he wishes to sign in at. Note that this scheme requires no certificate authority infrastructure.

There is an alternative solution to the key proliferation problem in symmetric challenge systems. The user can take advantage of a Snap2Pass-enabled OpenID provider and benefit from OpenID’s single sign-on properties across multiple web sites. Thus, the user’s Snap2Pass application needs to maintain a single shared secret between the user and his OpenID provider. The number of keys is limited to the number of OpenID providers he uses. He may even use the same private/public key pair across his OpenID providers enabling Snap2Pass to maintain fewer keys.

3.3 Phone Theft and Key Revocation

If a phone is lost or stolen, that phone can potentially be used to impersonate the user at all websites where the user has an account. Snap2Pass mitigates this issue in two ways.

Firstly, the Snap2Pass application can require the user to authenticate to the phone before the application can be used. Rather than implement an unlock feature in Snap2Pass we rely on the phone’s locking mechanism for this purpose. Users who worry about device theft can configure their phones to require a pass code before applications like Snap2Pass can be launched. This forces a thief to first override the phone’s locking mechanism. Moreover, several

phone vendors provide a remote kill feature that destroys data on the phone in case it is lost or stolen.

Secondly, when a phone is lost, users can easily revoke the Snap2Pass credentials on the phone by visiting web sites where they have an account and resetting their Snap2Pass credentials at those sites. This results in a new keying material generated for the user thus invalidating the secrets on the lost phone.

4. Security Analysis of Snap2Pass

We describe a number of attacks on the system and how they are addressed. Throughout the section we assume that the login process and the subsequent session on the PC are served over SSL so that basic session hijacking (i.e. the attacker waits for authentication to complete and then hijacks the session) is not possible.

We first observe that with Snap2Pass, unlike passwords, a compromise at one web site does not affect the user’s account at other sites. To see why, recall that in the symmetric scheme Snap2Pass maintains a different shared secret with each site. In the public-key scheme, the site never stores the phone’s secret key. Thus, in neither case does a compromise of one site affect another.

It is also worth noting that since the user never types in their password, Snap2Pass protects users against present day keylogging malware installed on the user’s PC. Nevertheless, more sophisticated malware on the user’s PC (e.g. [6]) can defeat Snap2Pass.

4.1 Offline Phishing

An offline phishing attack refers to a phisher who sets up a static spoofed web site and then waits for users to authenticate at the site. The term “offline” refers to the fact

that the phisher scrapes the target web site's login page offline. For sites using password authentication, an offline phisher obtains a list of username/password that can be sold to others. We note that users who fall victim to this attack typically ignore information displayed in the address bar [5]. Consequently, the SSL lock icon or the extended validation colors in the address bar do not prevent this attack.

Snap2Pass clearly prevents offline phishing since the phisher does not obtain a credential that can be used or sold. In fact, the offline phisher gets nothing since the phone sends its response directly to the target web site. Recall that during account creation Snap2Pass records the target web site's address on the phone. During login, it sends the response to that address. Consequently, the offline phisher will never see the response.

4.2 Online Phishing and Active Man in the Middle

In an online phishing attack, the attacker creates a spoofed web site that constantly scrapes the target web site. The phisher lures users to the spoofed site and uses their responses to immediately login to their account at the target site. Once in, the phisher can take any action on the user's account. This attack easily defeats one-time password mechanisms and many phishing toolkits now work this way [4].

This attack poses some risk to Snap2Pass. The phisher could mimic a browser to the target site and masquerade as a server to the unsuspecting user. When a user visits the phishing site, the phisher retrieves a current QR code from the target site and sends it to the user's PC browser. The user's phone processes the QR code and sends the correct response to the target site. Consequently the phisher's browser is logged into the user's account at the target site. As in the offline phishing case, we cannot rely on security indicators in the browser chrome to alert the user to this attack.

Snap2Pass defends against this attack using geolocation information. Recall that in Snap2Pass the target web site communicates with the user's PC and with the user's phone. In normal use, the two are in close proximity. In an online phishing attack, the site communicates with the phishing server and the user's phone. The two are very likely to be far apart. Thus, the web site can use geolocation information to test if the two IP addresses it is seeing are in close proximity. If so, it allows the connection and if not it rejects it. Thus, for the phisher to succeed he must find a compromised host close to the victim user and place the phishing server there. While not impossible, in most phishing settings, this will be quite challenging for the phisher.

It is worth noting that this attack is easily defeated using a PC browser extension. The extension would retrieve the SSL session key used in the connection to the web site (i.e. the phishing site) and embed a hash of this key in the QR code (if the connection is in the clear the data field would

be empty). The phone would send this hash to the real site along with its response to the challenge. The web site would now see that the browser's SSL key (used to communicate with the frontend of the phishing server) is different from its own SSL key (used to communicate with the backend of the phishing server) and would conclude that a man in the middle is interfering with the connection. We chose to not implement this defense since Snap2Pass is designed to work with existing unmodified browsers.

4.3 Signing Out

It is difficult for a web site to know if a user has walked away from an authenticated session [14]. With Snap2Pass we can use the phone as a proximity sensor, powered by the device's location sensors or accelerometers. For example, when the phone detects motion after authentication on the PC completes, it notifies the site. The site can then require re-authentication for subsequent requests. Thus, upon leaving an internet café, the user's session is immediately terminated. For web users on a moving train, the site may request one re-authentication and subsequently ignore motion notifications from the phone for the duration of the session.

5. Deployment of Snap2Pass

Section 3 describes the core Snap2Pass algorithm for logging onto one website. We now describe how we use a single mobile Snap2Pass client to log onto multiple websites, potentially with multiple personas. We also describe how by leveraging OpenID, we can enable the adoption of this technology immediately across a large number of existing websites.

5.1 Multiple Web Sites

In practice, we wish to carry only one Snap2Pass client on our phone to log onto multiple websites. The Snap2Pass client maintains a mapping from providers to accounts. It may also maintain multiple accounts per provider, allowing the user to select their desired identity during a login attempt. The response message from the phone to the web site contains the user's identity that is logging in along with the corresponding cryptographic response.

The biggest risk in extending to multiple web sites is a greater exposure to online phishing attacks. Now, a user has become accustomed to logging in to a variety of sites with the same mobile application, and they must be aware of the site at which they are authenticating. We associate a recognizable image with the web site and that image is displayed on the phone during login, as illustrated in Figure 5. The phone obtains the image at account creation time. Furthermore, recall that the cryptographic hash used to compute the response contains server information, so that the web site is certain that the mobile client was made aware of the correct login target.



Figure 5. Confirmation of a Login

5.2 OpenID

We have implemented Snap2Pass as a custom OpenID provider. Many web sites today have adopted the use of OpenID, enabling single sign-on using their OpenID credentials. The key advantage is that all of the websites that support OpenID can enable Snap2Pass based login without requiring any code changes on their end. The user's credentials reside with an OpenID provider that uses Snap2Pass. We used Snap2Pass to log into several websites supporting the standard, including: Slashdot, ProductWiki, ccMixter, and LiveJournal. Upon typing in an OpenID account name to the web site of a relying party, the web page automatically redirects the user to the login page of the OpenID provider. In this case, our OpenID provider presents the user with the Snap2Pass QR code, and the login process proceeds as described in Section 3. Once the login process completes, the OpenID provider signals the result to the relying party web site.

6. Snap2Web: Synergy Between the Phone and the PC Browser

We now turn to other applications of securely pairing a PC browser with a phone. The system, called Snap2Web, includes a number of applications that combine the best of both platforms.

6.1 User Experience

To start Snap2Web, we simply visit a web page hosting the Snap2Web service and snap the displayed QR code which contains all of the information needed to create an interactive session:

```
{ protocol: "V1"
  , host: "snap2.stanford.edu"
  , session-id: "3918a37dc9f5b"
}
```

With this scan, the PC browser and phone are paired and can exchange messages by way of the server. The user can

break this connection via the logout button on the mobile client or simply by closing the browser window.

Many different forms of cooperation can be provided easily once the browser and the phone are paired. Here are some functionalities we have developed:

- Making phone calls. Once the connection is established, the PC browser pulls the user's contacts, making them available in-browser. We can use the generous display and input devices on our PC to manage actions on our phone. We can quickly browse and search our contacts, and with a click can dial them on the mobile. We can also enter a phone number directly into a browser text box for dialing. Initiating calls from the PC browser can be helpful when using a headset so that there is no further need to manipulate the device directly.
- SMS messaging. Instead of typing SMS messages on the cell phone, we use the PC's keyboard to enter messages and send them from our mobile phone number. The phone's received messages are displayed on the PC browser so that the entire conversation is visible on the PC. Several messaging threads can be displayed at once.
- Transferring webpages. A web page can be transferred from the phone's browser to the desktop using Android's "Share" dialog. Thus, with a press of a button, a web page is moved from the phone to the PC.
- Keyboard entry. Snap2Web extends the cut-and-paste metaphor across the phone and PC. An in-browser text box allows text to be sent from the PC directly to the mobile's clipboard. Once on the clipboard, the text can be inserted in any application with a standard text field.

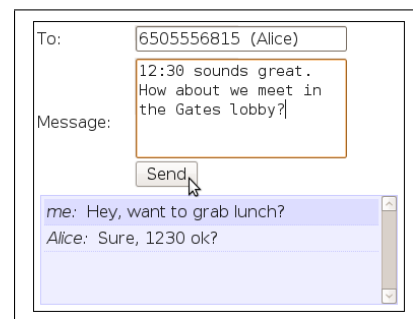


Figure 6. The Snap2Web in-browser SMS utility.

6.2 Snap2Web With a Trusted Server

Because a cell phone is not typically addressable by a browser, a trusted server, if one is available, can be used as a rendezvous point. In our implementation, this node is a standard XMPP server. When the browser page is loaded, it creates an XMPP chat room, identified by a random nonce. The browser then encodes the connection information in a QR code for receipt by the mobile device. The browser

and mobile both communicate with the server using standard SSL. The server serves as a trusted relay between the browser and the cell phone. Once this connection is established between the browser and the phone, all of the above interactions are provided easily.

6.3 Snap2Web With an Untrusted Server

It is also possible to implement Snap2Web without exposing the contents of the interaction to a trusted relay server. The PC browser generates both a random session-id for identifying the communication session and a secret session-key used for message level encryption. The browser sends the session-id to the server, but keeps the session-key to itself. Moreover, the browser generates a QR code containing the session-id and session-key. Once the phone scans the QR code, the PC browser and phone have a shared session-key unknown to the server which the use to encrypt and authenticate messages between them.

QR codes are capable of encoding all the information needed, including the 128-bit cryptographic key. A version 4 code, for example, can store 114 alphanumeric characters [12] and is readily decodable by a modern smartphone.

Since Snap2Web works with an unmodified PC web browser, all PC-side code is implemented in Javascript. This implementation is made simpler by available Javascript cryptographic libraries [15]. The code can be loaded onto the browser as a bookmarklet by visiting a trusted server.

7. Experimental Results

7.1 Implementation of Snap2Pass

Our implementation of Snap2Pass includes server-side code, called a provider, and a mobile client. The provider and the client provide a reference implementation for the server and client ends of the Snap2Pass protocol, respectively.

7.1.1 Provider

The provider is implemented as a custom OpenID provider and offers server-side challenge/response functionality as described above. OpenID is a popular protocol for federated identity management and single sign-on. With the addition of a layer of indirection, it enable tens of thousands of existing OpenID consumer web sites to use Snap2Pass without requiring modification of their server-side login protocols.

The provider implementation makes use of the Joid open source project, and is written in Java. It is loosely coupled to Joid; thus it can be plugged into other standard OpenID providers. The custom provider consists of a symmetric-key based challenge response system, account management and a web portal. The challenge response modules are written in Java using built-in crypto libraries. It includes modules for symmetric key generation, and HMAC-based challenge/response creation and verification. The account management modules manage user accounts, provide persis-

tence and include a cache for fast lookup of incoming responses.

The web portal adds QR code features to the OpenID provider. It includes custom registration and login pages, implemented as Java Server Pages (JSP) to support the Snap2Pass account creation and login protocol. On completion of the login protocol, the web portal integrates with the provider backend to signal the result using the OpenID protocol. This enables existing OpenID consumer sites to support Snap2Pass with no code changes.

The provider module has approximately 1,600 source lines of code (SLOC).

7.1.2 Mobile Client

The mobile client is written in Java for the Android environment. It implements the client-side Snap2Pass protocol, and offers functionality for credential management and symmetric key challenge/response computation. We use Android's SharedPreferences API to store and manage credentials retrieved from the provider. In a production implementation, the credentials are managed using a secure credential manager. The login module uses built-in APIs to compute responses to challenges. We use Android's intent system and the ZXing project to scan and consume QR codes. For improved security, the scanning functionality will be embedded directly in the application. The mobile client has approximately 400 SLOC.

7.2 User Study of Snap2Pass

We compare our implementation against existing secure authentication mechanisms. Our goal is to provide a system that is less cumbersome to use than what is currently available without hurting security. There are two main usability concerns here: that Snap2Pass is intuitive enough to be used by non-technical users and that frequent logins are as fast and effortless as possible.

7.2.1 Using QR Codes

We presented our approach to new users to see their reaction. Our subjects ranged from savvy smartphone users to those who are considerably less technical.

We found that the UI design for pinpointing a QR code is important. The visual feedback on the phone's screen was helpful in guiding the user to a successful login. Sometimes, a user would believe a code should be recognizable by the device before the software is able to locate it, causing some confusion on early logins. Subsequent uses were much faster and free of this confusion. Usually two or three sample trials were enough for the user to become comfortable with the system. Most users, including the non-technical, agree that the approach is "very simple" and usable.

7.2.2 Comparison with RSA SecurID and CRYPTOCARD

We compared the login times required for Snap2Pass with the times for other authentication systems. Our focus was on comparing the amount of time required for an experienced user to complete a successful authentication. We compared our system with two others in active deployment: RSA SecurID and CRYPTOCARD. We simulated the logins by providing phone interfaces for all three protocols.

For SecurID, our workflow was similar to a standard username/password login, with the addition of a “token” field. This token was generated on the mobile device, and the generation would be synchronized with the authentication server in deployment. Our CRYPTOCARD simulation involved an extra step. First, the user submits his username to the authentication server. The server responds with a random 6-character hex string, which the user must enter into his mobile device. The mobile device then generates a 6-character hex response, which the user inputs into the web form, along with his password. In our scheme, a user visits the authentication page, which presents a QR code. The user runs the “Log In” routine on his phone and scans the QR code, completing the authentication.

For each scheme, we issued twenty logins and recorded the amount of wall clock time required, from the time the login page loaded to the time the secured content was displayed. We assumed that the second factor device was at hand and the required application was launchable with one button press. Figure 7 shows the results of the user study. The error bars indicated were computed by the standard formula $\frac{\sigma}{\sqrt{n}}$, where σ is the standard deviation and n is the number of inputs.

The study shows that an experienced user takes about 7 seconds to log in using Snap2Pass. Overall, Snap2Pass was approximately 2.5 times faster than the CRYPTOCARD system, and 30% faster than RSA tokens. Most of the time required for logging in to both the SecurID and CRYPTOCARD systems was spent inputting data into forms, and copying values between devices—a process that is limited by the human’s typing and reading abilities. For Snap2Pass, most of the time was machine bounded. Our application required approximately two seconds on average to load the phone’s camera module from an idle state. Most of the remaining time was spent in the ZXing library trying to locate and decode the QR code. A visually apparent challenge for the library was auto-adjusting the aperture to read information from an active display as opposed to a passive one. These bottlenecks can be improved upon both in software and with upgraded device profiles, as opposed to end-user efforts.

Both the SecurID and CRYPTOCARD systems involve human-entered form data, a process that is prone to errors. In our user study, one out of twenty logins for both systems resulted in a failed login, essentially doubling the amount of time required to log in. Once the user has authenticated with his phone, our scheme does not suffer this shortcoming, as

all data is parsed, calculated, and submitted by the devices directly.

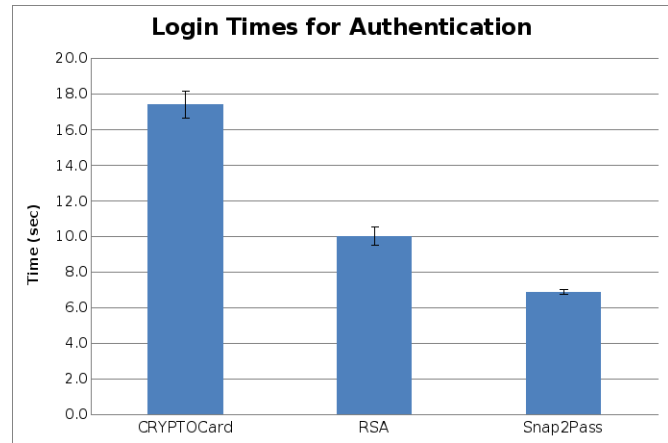


Figure 7. Average login time required across three secure authentication schemes.

7.3 User study of Snap2Web

To demonstrate the utility of Snap2Web, we ran a user study on its SMS capabilities. We were interested in the time required to send an SMS from a mobile phone versus from our web interface. Our assumption is that the Snap2Web web application would be available to the user as an already-open tab in their browser window. This pairing can be established in a few seconds, as demonstrated in our Snap2Pass user study.

We measured the amount of time required to send an SMS across three platforms—a mobile phone using the T9 entry mode, a smartphone with a full QWERTY keyboard, and our Snap2Web’s SMS feature. Our task was to select a contact and send the message “The quick brown fox jumps over the lazy dog.” Our results over twenty trials are shown in Figure 8. In our study, T9 entry was slightly faster than the mobile QWERTY, and our Snap2Web system was nearly twice as fast as the smartphone QWERTY entry.

8. Related Work

8.1 Mobile/Web Authentication

Using cell phone for authenticating people is an old idea. Both Wu et al. [16] and Parno et al. [10] use phones to carry user credentials and to help establish a secure channel to a remote site. Aloul and Zahidi discuss the use of one time passwords generated on a mobile phone for use with ATMs [1]. At the time these results were published, it was not possible to build a system like Snap2Pass on a cell phone. By using modern smart phones, we are able to provide a system that is easier to use without compromising security.

Bellovin et al. [2] propose EKE, a protocol for shared key exchange, which has been refined further. In contrast,

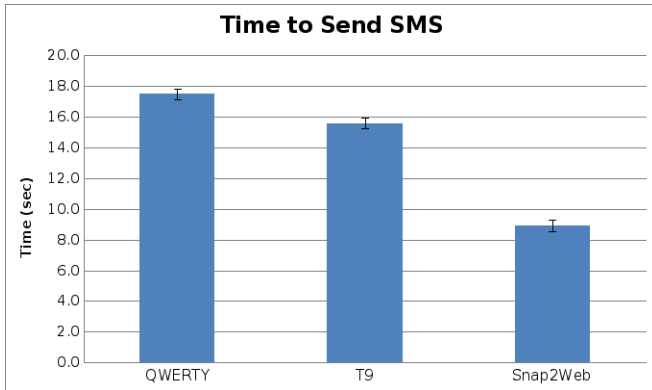


Figure 8. Average time required to send an SMS across three platforms.

we rely on QR codes over SSL to transfer the shared secret between the PC browser and the phone.

8.2 Device Pairing

Mccune et al. [9] have previously combined phone cameras and two dimensional barcodes for transmitting public keys from one device to another. Other clever approaches to device pairing use the accelerometer for generating a shared key [7, 8] or for proving proximity [3].

In [11], Pierce et al. explore the possible uses of pairing a mobile with a PC, presenting several useful utilities. Their model uses a centrally managed account to discover services, as opposed to our ad-hoc pairing technique.

9. Conclusions

We described Snap2Pass, an easy to use authentication system that defeats many of the attacks on traditional password schemes on the Web. Snap2Pass is implemented as a custom OpenID provider, thereby immediately enabling usage on the tens of thousands of websites that accept OpenID-based authentication, without any server-side code changes. Our user studies show that the system is fast to use and easy to learn.

We also described Snap2Web, a system that leverages a quick and secure pairing among a server, phone, and browser (called Snap2) to portal between the phone and the web browser. We demonstrated several applications like launching one-click phone calls from the PC, sending SMS messages on the phone from the PC, and transferring a web page from the phone to the PC. Finally, we show that the cut-and-paste paradigm can be extended bi-directionally between the PC and the phone. Our user study confirms the convenience of this service.

Both Snap2Pass and Snap2Web use an off the shelf PC browser with no modifications. Consequently, both systems work well with all popular browsers today.

We have an open-source implementation of Snap2Pass at <http://snap2.stanford.edu>. At the same site, users

can find Snap2Web where they can try out some of the applications described in the paper.

References

- [1] F. Aloul and S. Zahidi. Two factor authentication using mobile phones.
- [2] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *IEEE SYMPOSIUM ON RESEARCH IN SECURITY AND PRIVACY*, pages 72–84, 1992.
- [3] Bump technologies. bumptechnologies.com.
- [4] M. Cova, C. Kruegel, and G. Vigna. There is no free phish: An analysis of "free" and live phishing kits. In *Proc. of 2nd Unix wOOT '08*, 2008.
- [5] R. Dhamija, D. Tygar, and M. Hearst. Why phishing works. In *Proc. of CHI 2006*, pages 581–590, 2006.
- [6] C. Jackson, D. Boneh, and J. Mitchell. Transaction generators: Root kits for the web. In *proceedings of the 2nd USENIX Workshop on Hot Topics in Security*, 2007.
- [7] D. Kirovski, M. Sinclair, and D. Wilson. The martini synch. Technical report, Microsoft Research Technical Report, MSR-TR-2007-123, 2007.
- [8] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In *5th International Conference on Pervasive Computing*, volume 4480 of LNCS, pages 144–161, 2007.
- [9] J. M. Mccune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *In IEEE Symposium on Security and Privacy*, pages 110–124, 2005.
- [10] B. Parno, C. Kuo, and A. Perrig. Phoolproof phishing prevention. In *Proceedings of the 10th International Conference on Financial Cryptography and Data Security (FC'06)*, 2006.
- [11] J. S. Pierce and J. Nichols. An infrastructure for extending applications' user experiences across multiple personal devices. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 101–110, New York, NY, USA, 2008. ACM.
- [12] code.google.com/apis/chart/types.html#qrcodes.
- [13] M. V. Rafter. A breakout year for openid, 2009. technology.inc.com/security/articles/200902/openID.html.
- [14] B. Schneier. Unauthentication, 2009. www.schneier.com/blog/archives/2009/09/unauthentication.html.
- [15] E. Stark, M. Hamburg, and D. Boneh. Symmetric cryptography in javascript. In *Annual Computer Security Applications Conference, 2009*. crypto.stanford.edu/sjcl.
- [16] M. Wu, S. Garfinkel, and R. Miller. Secure web authentication with mobile phones. In *DIMACS Workshop on Usable Privacy and Security Software*, 2004.